

GESTIÓN DE LA CONFIGURACIÓN  
DEL SOFTWARE (GCS/SCM)\*

**C**UANDO se construye software de computadora, los cambios son inevitables. Además, los cambios aumentan el grado de confusión entre los ingenieros del software que están trabajando en el proyecto. La confusión surge cuando no se han analizado los cambios antes de realizarlos, no se han registrado antes de implementarlos, no se les han comunicado a aquellas personas que necesitan saberlo o no se han controlado de manera que mejoren la calidad y reduzcan los errores. Babich [BAB86] se refiere a este asunto cuando dice:

El arte de coordinar el desarrollo de software para minimizar...la confusión, se denomina gestión de configuración. La gestión de configuración es el arte de identificar, organizar y controlar las modificaciones que sufre el software que construye un equipo de programación. La meta es maximizar la productividad minimizando los errores.

La gestión de configuración del software (GCS) es una actividad de autoprotección que se aplica durante el proceso del software. Como el cambio se puede producir en cualquier momento, las actividades de GCS sirven para (1) identificar el cambio, (2) controlar el cambio, (3) garantizar que el cambio se implementa adecuadamente y (4) informar del cambio a todos aquellos que puedan estar interesados.

Es importante distinguir claramente entre el mantenimiento del software y la gestión de configuración del software. El mantenimiento es un conjunto de actividades de ingeniería del software que se producen después de que el software se haya entregado al cliente y esté en funcionamiento. La gestión de configuración del software es un conjunto de actividades de seguimiento y control que comienzan cuando se inicia el proyecto de ingeniería del software y termina sólo cuando el software queda fuera de la circulación.

## VISTAZO RÁPIDO

**¿Qué es?** Cuando construimos software de computadora, surgen cambios. Debido a esto, necesitamos controlarlos eficazmente. La gestión de la configuración del software (GCS) es un conjunto de actividades diseñadas para controlar el cambio identificando los productos del trabajo que probablemente cambien, estableciendo relaciones entre ellos, definiendo mecanismos para gestionar distintas versiones de estos productos, controlando los cambios realizados, y auditando e informando de los cambios realizados.

**¿Quién lo hace?** Todos aquellos que estén involucrados en el proceso de ingeniería de software están relacionados con la GCS hasta cierto punto, pero las posiciones de mantenimiento especializadas son creadas a veces para la gestión del proceso de GCS.

**¿Por qué es importante?** Si no controlamos el cambio, él nos controlará a nosotros. Y esto nunca es bueno. Es muy fácil para un flujo de cambios incontrolados llevar al caos a un proyecto de software correcto. Por esta razón, la GCS es una parte esencial de una buena gestión del proyecto y una práctica formal de la ingeniería del software.

**¿Cuáles son los pasos?** Puesto que muchos productos de trabajo se obtienen cuando se construye el software, cada uno debe estar identificado unívocamente. Una vez que esto se ha logrado, se pueden establecer los mecanismos para el control del cambio y de las versiones. Para garantizar que se mantiene la calidad mientras se realizan los cambios, se audita el proceso, y para asegurar que aquellos

que necesitan conocer los cambios son informados, se realizan los informes.

**¿Cuál es el producto obtenido?** El Plan de Gestión de la Configuración del Software define la estrategia del proyecto para la GCS. Además, cuando se realiza la GCS formal, el proceso de control del cambio provoca peticiones de cambio del software e informes de órdenes de cambios de ingeniería.

**¿Cómo puedo estar seguro de que lo he hecho correctamente?** Cuando cualquier producto de trabajo puede ser estimado para ser supervisado y controlado: cuando cualquier cambio pueda ser seguido y analizado; cuando cualquiera que necesite saber algo sobre algún cambio ha sido informado, lo habremos realizado correctamente.

\* En inglés, Software Configuration Management.

## 9.1 GESTIÓN DE LA CONFIGURACIÓN DEL SOFTWARE

El resultado del proceso de ingeniería del software es una información que se puede dividir en tres amplias categorías: (1) programas de computadora (tanto en forma de código fuente como ejecutable), (2) documentos que describen los programas de computadora (tanto técnicos como de usuario) y (3) datos (contenidos en el programa o externos a él). Los elementos que componen toda la información producida como parte del proceso de ingeniería del software se denominan colectivamente configuración del software.

A medida que progresa el proceso del software, el número de *elementos de configuración del software* (ECSs) crece rápidamente. Una especificación del sistema produce un plan del proyecto del software y una especificación de requisitos del software (así como otros documentos relativos al hardware). A su vez, éstos producen otros documentos para crear una jerarquía de información. Si simplemente cada ECS produjera otros ECSs, no habría prácticamente confusión. Desgraciadamente, en el proceso entra en juego otra variable —el cambio—. El cambio se puede producir en cualquier momento y por cualquier razón. De hecho, la Primera Ley de la Ingeniería de Sistemas [BER80] establece: Sin importar en qué momento del ciclo de vida del sistema nos encontremos, el sistema cambiará y el deseo de cambiarlo persistirá a lo largo de todo el ciclo de vida.



No hay nada permarr ente excepto el cambio  
Heráclito, 500 AdC.

¿Cuál es el origen de estos cambios? La respuesta a esta pregunta es tan variada como los cambios mismos. Sin embargo, hay cuatro fuentes fundamentales de cambios:

- nuevos negocios o condiciones comerciales que dictan los cambios en los requisitos del producto o en las normas comerciales;
- nuevas necesidades del cliente que demandan la modificación de los datos producidos por sistemas de información, funcionalidades entregadas por productos o servicios entregados por un sistema basado en computadora;
- reorganización o crecimiento/reducción del negocio que provoca cambios en las prioridades del proyecto o en la estructura del equipo de ingeniería del software;
- restricciones presupuestarias o de planificación que provocan una redefinición del sistema o producto.

La gestión de configuración del software (GCS) es un conjunto de actividades desarrolladas para gestionar los cambios a lo largo del ciclo de vida del software de computadora.



*La mayoría de los cambios son justificados. No lamente los cambios. Mejor dicho, esté seguro que tiene los mecanismos preparados para realizarlos.*

### 9.1.1. Líneas base

Una línea base es un concepto de gestión de configuración del software que nos ayuda a controlar los cambios sin impedir seriamente los cambios justificados. La IEEE (Estándar IEEE 610.12-1990) define una línea base como:

Una especificación o producto que se ha revisado formalmente y sobre los que se **ha** llegado a un acuerdo, y que de ahí en adelante sirve como base para un desarrollo posterior y que puede cambiarse solamente a través de procedimientos formales de control de cambios.

Una forma de describir la línea base es mediante una analogía:

Considere las puertas de la cocina en un gran restaurante. Para evitar colisiones, una puerta está marcada como **SALIDA** y la otra como **ENTRADA**. Las puertas tienen topes que hacen que sólo se puedan abrir en la dirección apropiada.

Si un camarero recoge un pedido en la cocina, lo coloca en una bandeja luego se da cuenta de que ha cogido un plato equivocado, puede cambiar el plato corrector rápida e informalmente antes de salir de la cocina.

Sin embargo, si abandona la cocina, le da el plato al cliente y luego se le informa de su error, debe seguir el siguiente procedimiento: (1) mirar en **la** orden de pedido si ha habido algún error; (2) disculparse insistentemente; (3) volver a la cocina por la puerta de **ENTRADA**; (4) explicar el problema, etc.



Un producto de trabajo de la ingeniería del software se convierte en una línea base, solamente después de haber sido revisado y aprobado.

Una línea base es análoga a la cocina de un restaurante. Antes de que un elemento de configuración de software se convierta en una línea base, el cambio se puede llevar a cabo rápida e informalmente. Sin embargo, una vez que se establece una línea base, pasamos, de forma figurada, por una puerta de un solo sentido. Se pueden llevar a cabo los cambios, pero se debe aplicar un procedimiento formal para evaluar y verificar cada cambio.

En el contexto de la ingeniería del software, definimos una *línea base* como un punto de referencia en el desarrollo del software que queda marcado por el envío de uno o más elementos de configuración del software y la aprobación del ECS obtenido mediante una revisión técnica formal (Capítulo 8). Por ejemplo, los ele-

mentos de una *Especificación de Diseño* se documentan y se revisan. Se encuentran errores y se corrigen. Cuando todas las partes de la especificación se han revisado, corregido y aprobado, la *Especificación de Diseño* se convierte en una línea base. Sólo se pueden realizar cambios futuros en la arquitectura del software (documentado en la *Especificación de Diseño*) tras haber sido evaluados y aprobados. Aunque se pueden definir las líneas base con cualquier nivel de detalle, las líneas base más comunes son las que se muestran en la Figura 9.1.



Asegúrese que la base de datos del proyecto se mantiene sobre un entorno controlado, centralizado.

La progresión de acontecimientos que conducen a una línea base está también ilustrada en la Figura 9.1. Las tareas de la ingeniería del software producen uno o más ECSs. Una vez que un ECS se ha revisado y aprobado, se coloca en una *base de datos del proyecto* (también denominada *biblioteca del proyecto* o *depósito de software*). Cuando un miembro del equipo de ingeniería del software quiere hacer modificaciones en un ECS de línea base, se copia de la base de datos del proyecto a un área de trabajo privada del ingeniero. Sin embargo, este ECS extraído puede modificarse sólo si se siguen los controles GCS (tratados más adelante en este capítulo). Las flechas punteadas de la Figura 9.1 muestran el camino de modificación de una línea base ECS.

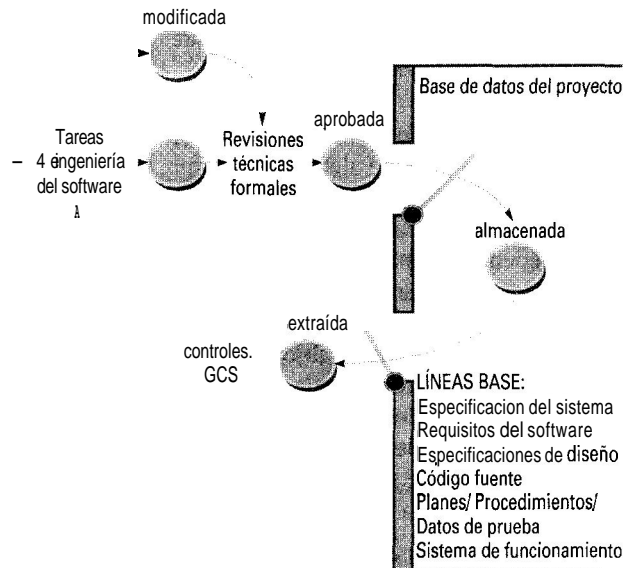


FIGURA 9.1. ECS de línea base y base de datos del proyecto.

## 9.1.2 Elementos de configuración del software

Ya hemos definido un elemento de configuración del software como la información creada como parte del proce-

so de ingeniería del software. Llevado al extremo, se puede considerar un ECS como una sección individual de una gran especificación o cada caso de prueba de un gran conjunto de pruebas. De forma más realista, un ECS es un documento, un conjunto completo de casos de prueba o un componente de un programa dado (p. ej., una función de C++ o un paquete de ADA).

En realidad, los ECSs se organizan como *objetos de configuración* que han de ser catalogados en la base de datos del proyecto con un nombre único. Un objeto de configuración tiene un nombre y unos atributos y está «conectado» a otros objetos mediante relaciones. De acuerdo con la Figura 9.2, los objetos de configuración, **Especificación de Diseño**, **modelo de datos**, **componente N**, **código fuente** y **Especificación de Prueba**, están definidos por separado. Sin embargo, cada objeto está relacionado con otros como muestran las flechas. Una flecha curvada representa una *relación de composición*. Es decir, **modelo de datos** y **componente N** son parte del objeto **Especificación de Diseño**. Una flecha recta con dos puntas representa una interrelación. Si se lleva a cabo un cambio sobre el objeto **código fuente**, las interrelaciones permiten al ingeniero de software determinar qué otros objetos (y ECSs) pueden verse afectados<sup>1</sup>.



Elementos de configuración del software.

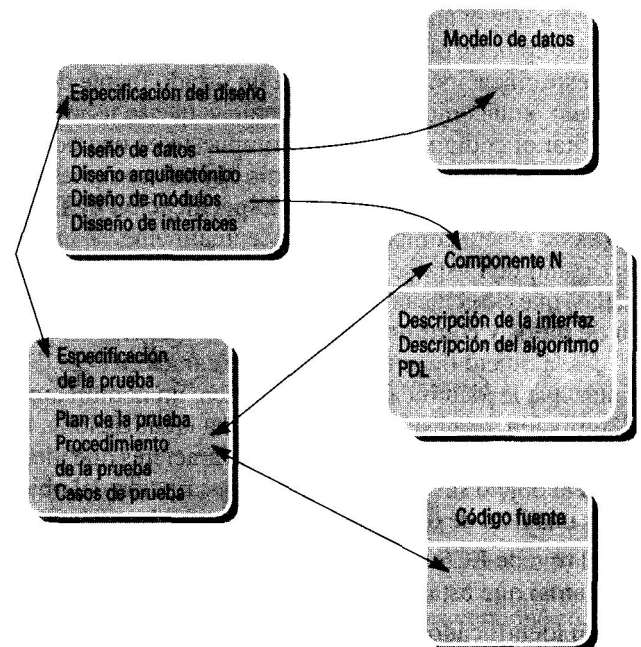


FIGURA 9.2. Objetos de configuración.

<sup>1</sup> Estas relaciones se tratan en la Sección 9.2.1 y la estructura de la base de datos del proyecto se trata con detalle en el Capítulo 31.

## 9.2 EL PROCESO DE GCS

La gestión de configuración del software es un elemento importante de garantía de calidad del software. Su responsabilidad principal es el control de cambios. Sin embargo, la GCS también es responsable de la identificación de ECSs individuales y de las distintas versiones del software, de las auditorías de la configuración del software para asegurar que se desarrollan adecuadamente y de la generación de informes sobre todos los cambios realizados en la configuración.

### Referencia Web

Las páginas amarillas de gestión de configuración contienen el listado más complejo de los recursos de GCS en la web. Para más información, consultar:  
[www.cs.colorado.edu/users/andre/tomconfiguration-management.html](http://www.cs.colorado.edu/users/andre/tomconfiguration-management.html)

Cualquier estudio de la GCS plantea una serie de preguntas complejas:

- ¿Cómo identifica y gestiona una organización las diferentes versiones existentes de un programa (y su documentación) de forma que se puedan introducir cambios eficientemente?
- ¿Cómo controla la organización los cambios antes y después de que el software sea distribuido al cliente?
- ¿Quién tiene la responsabilidad de aprobar y de asignar prioridades a los cambios?
- ¿Cómo podemos garantizar que los cambios se han llevado a cabo adecuadamente?
- ¿Qué mecanismo se usa para avisar a otros de los cambios realizados?

Estas cuestiones nos llevan a la definición de cinco tareas de GCS: *Identificación, control de versiones, control de cambios, auditorías de configuración y generación de informes.*

## 9.3 IDENTIFICACIÓN DE OBJETOS EN LA CONFIGURACIÓN DEL SOFTWARE

Para controlar y gestionar los elementos de configuración, se debe identificar cada uno de forma Única y luego organizarlos mediante un enfoque orientado a objetos. Se pueden identificar dos tipos de objetos [CHO89]: *objetos básicos* y *objetos compuestos*<sup>2</sup>. Un objeto básico es una «unidad de texto» creado por un ingeniero de software durante el análisis, diseño, codificación o pruebas. Por ejemplo, un objeto básico podría ser una sección de una especificación de requisitos, un listado fuente de un módulo o un conjunto de casos prueba que se usan para ejercitar el código. Un objeto compuesto es una colección de objetos básicos y de otros objetos compuestos. De acuerdo con la Figura 9.2, **la Especificación de Diseño** es un objeto compuesto. Conceptualmente, se puede ver como una lista de referencias con nombre (identificadas) que especifican objetos básicos, tales como **modelo de datos** y **componente N**.

Cada objeto tiene un conjunto de características distintas que le identifican de forma Única: un nombre, una descripción, una lista de recursos y una «realización». El nombre del objeto es una cadena de caracteres que identifica al objeto sin ambigüedad. La descripción del objeto es una lista de elementos de datos que identifican:

- el tipo de ECS (por ejemplo: documento, programa, datos) que está representado por el objeto;
- un identificador del proyecto;
- la información de la versión y/o el cambio;

<sup>2</sup> Como mecanismos para representar una versión completa de la configuración del software se ha propuesto el concepto de objeto agregado [GUS89]

### CLAVE

Las relaciones establecidos entre los objetos de configuración permiten al ingeniero del software evaluar el impacto del cambio.

Los recursos son «entidades que proporciona, procesa, referencia o son, de alguna otra forma, requeridas por el objeto». [CHO89], por ejemplo, los tipos de datos, las funciones específicas incluso los nombres de las variables pueden ser considerados recursos de objetos. La realización es una referencia a la «unidad de texto» para un objeto básico y nulo para un objeto compuesto.

La identificación del objeto de configuración también debe considerar las relaciones existentes entre los objetos identificados. Un objeto puede estar identificado como **<parte-de>** un objeto compuesto. La relación **<parte-de>** define una jerarquía de objetos. Por ejemplo, utilizando esta sencilla notación

Diagrama E-R 1.4 **<parte-de>** modelo de datos;  
 Modelo de datos **<parte-de>** especificación de diseño;

creamos una jerarquía de ECSs.

No es realista asumir que la Única relación entre los objetos de la jerarquía de objetos se establece mediante largos caminos del árbol jerárquico. En muchos casos, los objetos están interrelacionados entre ramas de la

jerarquía de objetos. Por ejemplo, el modelo de datos está interrelacionado con los diagramas de flujo de datos (suponiendo que se usa el análisis estructurado) y también está interrelacionado con un conjunto de casos de prueba para una clase particular de equivalencia. Las relaciones a través de la estructura se pueden representar de la siguiente forma:

Modelo de datos <interrelacionado> modelo de flujo de datos;  
Modelo de datos <interrelacionado> caso de prueba de la clase m;

#### Referencia cruzada

Los modelos de datos y los diagramas de flujo de datos se tratan en el Capítulo 12.

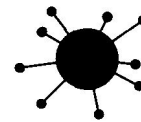
En el primer caso, la interrelación es entre objetos compuestos, mientras que en el segundo caso, la relación es entre un objeto compuesto (**modelo de datos**) y un objeto básico (**caso de prueba de la clase m**).

Las interrelaciones entre los objetos de configuración se pueden representar con un *lenguaje de interconexión de módulos (LIM)* [NAR87]. Un LIM describe las interdependencias entre los objetos de configuración y permite construir automáticamente cualquier versión de un sistema.

El esquema de identificación de los objetos de software debe tener en cuenta que los objetos evolucionan a lo largo del proceso de ingeniería del software. Antes de que un objeto se convierta en una línea base, puede cambiar varias veces, e incluso cuando ya es una línea base, los cambios se presentan con bastante frecuencia. Se puede crear un *grafo de evolución*

[GUS89] para cualquier objeto. El grafo de evolución describe la historia de los cambios de un objeto y aparece ilustrado en la Figura 9.3. El objeto de configuración 1.0 pasa la revisión y se convierte en el objeto 1.1. Algunas correcciones y cambios mínimos producen las versiones 1.1.1 y 1.1.2, que van seguidas de una actualización importante, resultando el objeto 1.2. La evolución de objeto 1.0 sigue hacia 1.3 y 1.4, pero, al mismo tiempo, una gran modificación del objeto produce un nuevo camino de evolución, la versión 2.0. A estas dos versiones se les sigue dando soporte.

Se pueden realizar cambios en cualquier versión, aunque no necesariamente en todas. ¿Cómo puede el desarrollador establecer las referencias de todos los componentes, documentos, casos de prueba de la versión 1.4.? ¿Cómo puede saber el departamento comercial los nombres de los clientes que en la actualidad tienen la versión 2.1.? ¿Cómo podemos estar seguros de que los cambios en el código fuente de la versión 2.1 han sido reflejados adecuadamente en la correspondiente documentación de diseño? Un elemento clave para responder a todas estas preguntas es la identificación.



Herramientas CASE-GCS

Se han desarrollado varias herramientas automáticas para ayudaren la tarea de identificación (y otras de GCS). En algunos casos, se diseña la herramienta para mantener copias completas de las versiones más recientes.

## 9.4 CONTROL DE VERSIONES

El control de versiones combina procedimientos y herramientas para gestionar las versiones de los objetos de configuración creados durante el proceso del software. Clemm [CLE89] describe el control de versiones en el contexto de la GCS:



El esquema que Vd. estableció para los ECS debería incorporar el número de versión.

La gestión de configuración permite a un usuario especificar configuraciones alternativas del sistema de software mediante la selección de las versiones adecuadas. Esto se puede gestionar asociando atributos a cada versión del software y permitiendo luego especificar [y construir] una configuración describiendo el conjunto de atributos deseado.

Los «atributos» que se mencionan pueden ser tan sencillos como un número específico de versión asociado a cada objeto o tan complejos como una cadena de variables lógicas (indicadores) que especifiquen

tipos de cambios funcionales aplicados al sistema [LIE89].

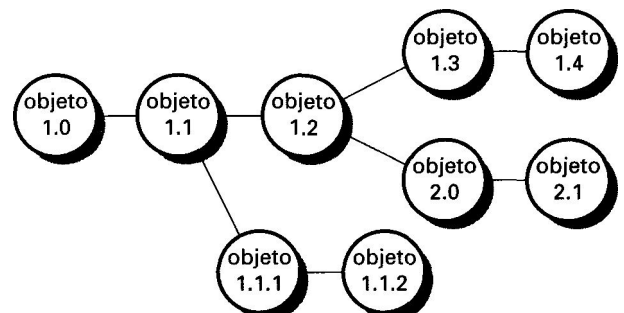


FIGURA 9.3. Grafo de evolución.

Una representación de las diferentes versiones de un sistema es el grafo de evolución mostrado en la Figura 9.3. Cada nodo del grafo es un objeto compuesto, es decir, una versión completa del software. Cada versión del software es una colección de ECSs (código fuente, documentos, datos) y cada versión puede estar com-

puesta de diferentes variantes. Para ilustrar este concepto, consideremos una versión de un sencillo programa que está formado por los componentes 1, 2, 3, 4 y 5<sup>3</sup>. El componente 4 sólo se usa cuando el software se implementa para monitores de color. El componente 5 se implementa cuando se dispone de monitor monocromo. Por tanto, se pueden definir dos variantes de la versión: (1) componentes 1, 2, 3 y 4; (2) componentes 1, 2, 3 y 5.

Para construir la **variante** adecuada de una determinada versión de un programa, a cada componente se le asigna una «tupla de atributos» —una lista de características que definen si se ha de utilizar el componente cuando se va a construir una determinada versión del software—. A cada variante se le asigna uno o más atributos. Por ejemplo, se podría usar un atributo color para definir qué componentes se deben incluir para soporte de monitores en color.

**Cita:**  
Cualquier cambio, incluso un cambio para mejor, está siempre acompañado por desventajas y disconformidades.  
Arnold Bennett

Otra forma de establecer los conceptos de la relación entre componentes, variantes y versiones (revisiones) es representarlas como un *fondo de objetos* [REI89]. De acuerdo con la Figura 9.4, la relación entre los objetos de configuración y los componentes, las variantes y las versiones se pueden representar como un espacio tridimensional. Un componente consta de una colección de

objetos del mismo nivel de revisión. Una variante es una colección diferente de objetos del mismo nivel de revisión y, por tanto, coexiste en paralelo con otras variantes. Una nueva versión se define cuando se realizan cambios significativos en uno o más objetos.

En la pasada década se propusieron diferentes enfoques automatizados para el control de versiones. La principal diferencia entre los distintos enfoques está en la sofisticación de los atributos que se usan para construir versiones y variantes específicas de un sistema y en la mecánica del proceso de construcción.

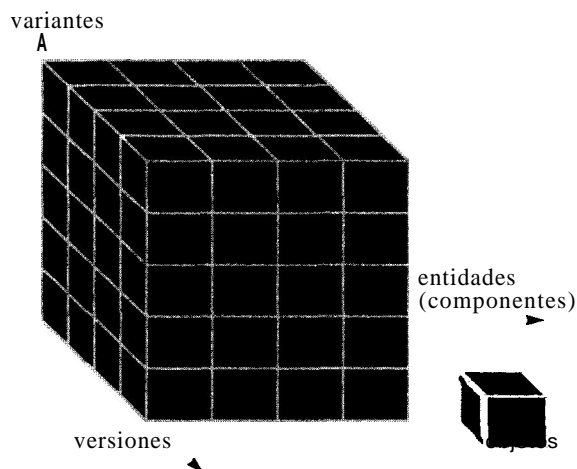


FIGURA 9.4. Representación en fondo de objetos de los componentes, variantes y versiones [REI89].

## 9.5 CONTROL DE CAMBIOS

La realidad del *control de cambio* en un contexto moderno de ingeniería de software ha sido bien resumida por James Bach [BAC98] :

El control de cambio es vital. Pero las fuerzas que lo hacen necesario también lo hacen molesto. Nos preocupamos por el cambio porque una diminuta perturbación en el código puede crear un gran fallo en el producto. Pero también puede reparar un gran fallo o habilitar excelentes capacidades nuevas. Nos preocupamos por el cambio porque un desarrollador pícaro puede hacer fracasar el proyecto; sin embargo las brillantes ideas nacidas en la mente de estos pícaros, y un pesado proceso de control de cambio pueden disuadirle de hacer un trabajo creativo.

Bach reconoce que nos enfrentamos a una situación a equilibrar. Mucho control de cambio y crearemos problemas. Poco, y crearemos otros problemas.

En un gran proyecto de ingeniería de software, el cambio incontrolado lleva rápidamente al caos. Para estos pro-

**Cita:**  
El arte del progreso está en mantener el orden en medio del cambio y mantener el cambio en medio del orden.  
Alfred North Whitehead

yectos, el control de cambios combina los procedimientos humanos y las herramientas automáticas para proporcionar un mecanismo para el control del cambio. El proceso de control de cambios está ilustrado esquemáticamente en la Figura 9.5. Se hace una *petición de cambio*<sup>4</sup> y se evalúa para calcular el esfuerzo técnico, los posibles efectos secundarios, el impacto global sobre otras funciones del sistema y sobre otros objetos de la configuración. Los resultados de la evaluación se presentan

<sup>3</sup> En este contexto, el término «componente» se refiere a todos los objetos compuestos y objetos básicos: un ESC de línea base. Por ejemplo, un componente de «entrada» puede estar constituido por seis componentes de software distintos, cada uno responsable de una subfunción de entrada.

<sup>4</sup> Aunque muchas de las peticiones de cambio se reciben durante la fase de mantenimiento, en este estudio tomamos un punto de vista más amplio. Una petición de cambio puede aparecer en cualquier momento durante el proceso del software.

como un informe de cambios a la *autoridad de control de cambios (ACC)* —una persona o grupo que toma la decisión final del estado y la prioridad del cambio—. Para cada cambio aprobado se genera una *orden de cambio de ingeniería (OCI)*. La OCI describe el cambio a realizar, las restricciones que se deben respetar y los criterios de revisión

y de auditoría. El objeto a cambiar es «dato de baja» de la base de datos del proyecto; se realiza el cambio y se aplican las adecuadas actividades de **SQA**. Luego, el objeto es «dato de alta» en la base de datos y se usan los mecanismos de control de versiones apropiados (Sección 9.4) para crear la siguiente versión del software.

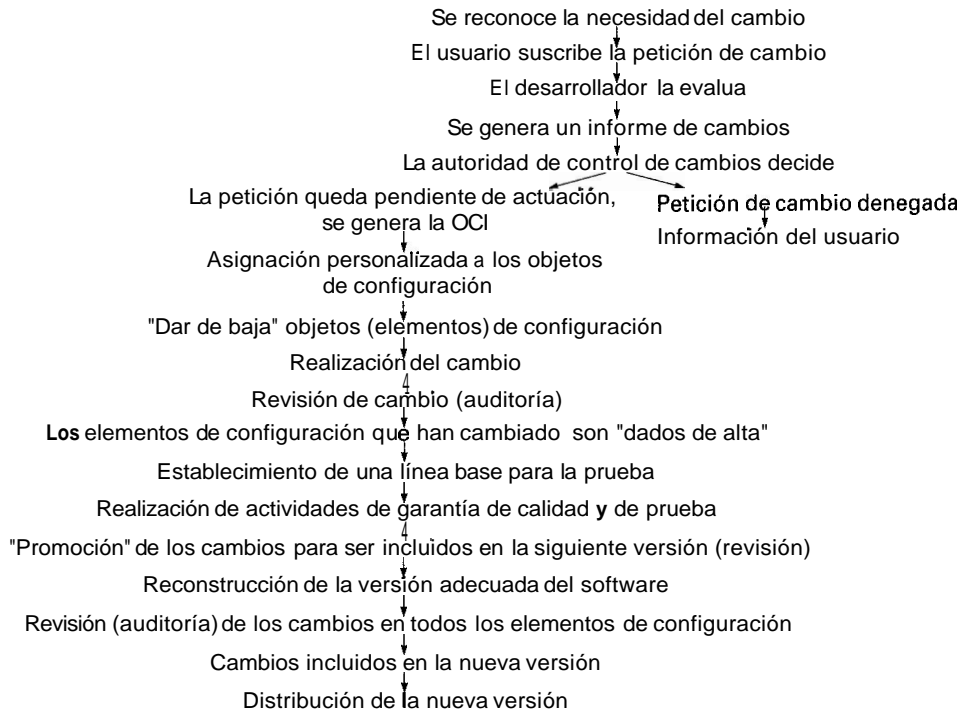


FIGURA 9.5. El proceso de control de cambios.

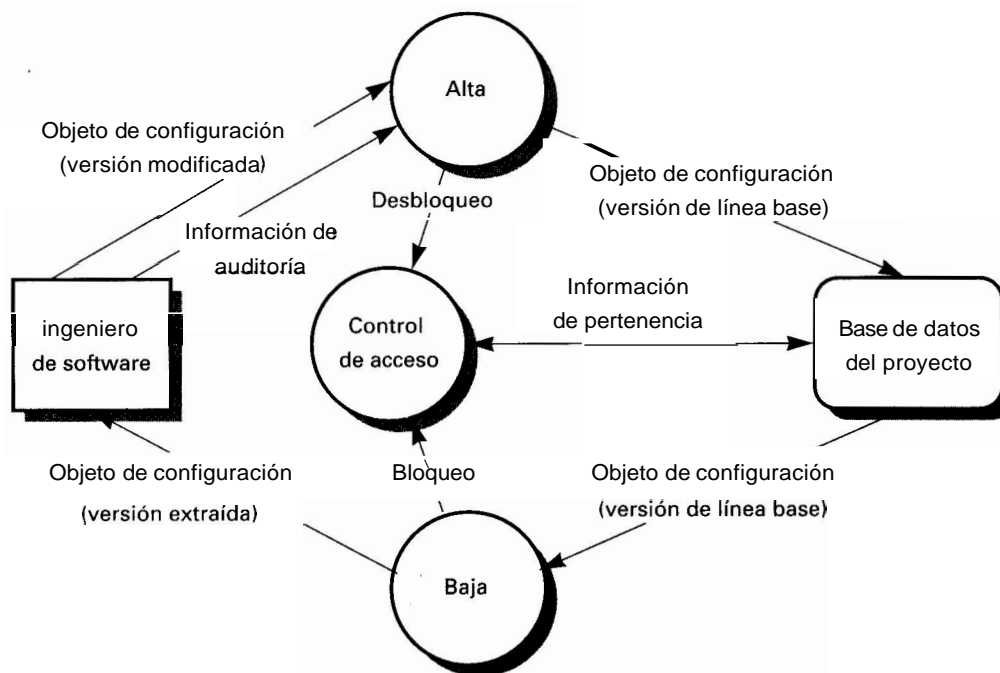


FIGURA 9.6. Control de acceso y de sincronización.



*La confusión conduce a los errores - algunas de ellas muy serias —. Los controles de acceso y de sincronización evitan la confusión. Implemente ambos, incluso si su enfoque tiene que ser simplificado para adaptarlo a su cultura de desarrollo.*

Los procesos de «alta» y «baja» implementan dos elementos importantes del control de cambios --control de acceso y control de sincronización—. El *control de acceso* gobierna los derechos de los ingenieros de software a acceder y modificar objetos de configuración concretos. El *control de sincronización* asegura que los cambios en paralelo, realizados por personas diferentes, no se sobreescriben mutuamente [HAR89].

El flujo de control de acceso y de sincronización están esquemáticamente ilustrados en la Figura 9.6. De acuerdo con una petición de cambio aprobada y una OCI, un ingeniero de software *da de baja* a un objeto de configuración. Una función de control de acceso comprueba que el ingeniero tiene autoridad para dar de baja el objeto, y el control de sincronización *bloquea* el objeto en la base de datos del proyecto, de forma que no se puedan hacer más actualizaciones hasta que se haya reemplazado con la nueva versión. Fíjese que se pueden dar de baja otras copias, pero no se podrán hacer otras actualizaciones. El ingeniero de software modifica una copia del objeto de línea base, denominada *versión extraída*. Tras la *SQA* y la prueba apropiada, se *da de alta* la versión modificada del objeto y se desbloquea el nuevo objeto de línea base.

Puede que algunos lectores empiecen a sentirse incómodos con el nivel de burocracia que implica el proceso anterior. Esta sensación es normal. Sin la protección adecuada, el control de cambios puede ralentizar el progreso y crear un papeleo innecesario. La mayoría de los desarrolladores de software que disponen de mecanismos de control de cambios (desgraciadamente la mayoría no tienen ninguno) han creado varios niveles de control para evitar los problemas mencionados anteriormente.



*Opte por un poco más de control de cambio del que pensaba necesitar en un principio. Es probable que mucha pueda ser la cantidad apropiada.*

Antes de que un ECS se convierta en una línea base, sólo es necesario aplicar *un control de cambios informal*. El que haya desarrollado el ECS en cuestión podrá hacer cualquier cambio justificado por el proyecto y por los requisitos técnicos (siempre que los cambios no impacten en otros requisitos del sistema más amplios que queden fuera del ámbito de trabajo del encargado del desarrollo). Una vez que el objeto ha pasado la revisión técnica formal y ha sido aprobado, se crea la línea base. Una vez que el ECS se convierte en una línea base, aparece el *control de cambios a nivel de proyecto*. Ahora, para hacer un cambio, el encargado del desarrollo debe recibir la aprobación del gestor del proyecto (si el cambio es «local») o de la ACC si el cambio impacta en otros ECSs. En algunos casos, se dispensa de generar formalmente las peticiones de cambio, los informes de cambio y las OCI. Sin embargo, hay que hacer una evaluación de cada cambio así como un seguimiento y revisión de los mismos.

Cuando se distribuye el producto de software a los clientes, se instituye el *control de cambios formal*. El procedimiento de control de cambios formal es el que aparece definido en la Figura 9.5.

La autoridad de control de cambios (ACC) desempeña un papel activo en el segundo y tercer nivel de control. Dependiendo del tamaño y de las características del proyecto de software, la ACC puede estar compuesta por una persona —el gestor del proyecto— o por varias personas (por ejemplo, representantes del software, del hardware, de la ingeniería de las bases de datos, del soporte o del departamento comercial, etc.). El papel de la ACC es el de tener una visión general, o sea, evaluar el impacto del cambio fuera del ECS en cuestión. ¿Cómo impactará el cambio en el hardware? ¿Cómo impactará en el rendimiento? ¿Cómo alterará el cambio la percepción del cliente sobre el producto? ¿Cómo afectará el cambio a la calidad y a la fiabilidad? La ACC se plantea estas y otras muchas cuestiones.



*El cambio es inevitable, excepto para las máquinas de venta.*  
**Bumper Sticker**

## 9.6 AUDITORÍA DE LA CONFIGURACIÓN

La identificación, el control de versiones y el control de cambios ayudan al equipo de desarrollo de software a mantener un orden que, de otro modo, llevaría a una situación caótica y sin salida. Sin embargo, incluso los mecanismos más correctos de control de cambios hacen un seguimiento al cambio sólo hasta que se ha generado la OCI. ¿Cómo podemos asegurar que el cambio se


ha implementado correctamente? La respuesta es doble: (1) revisiones técnicas formales y (2) auditorías de configuración del software.

Las revisiones técnicas formales (presentadas en detalle en el Capítulo 8) se centran en la corrección técnica del elemento de configuración que ha sido modificado. Los revisores evalúan el ECS para determinar la con-



sistencia con otros ECSs, las omisiones o los posibles efectos secundarios. Se debe llevar a cabo una revisión técnica formal para cualquier cambio que no sea trivial.

Una *auditoría de configuración del software* complementa la revisión técnica formal al comprobar características que generalmente no tiene en cuenta la revisión. La auditoría se plantea y responde las siguientes preguntas:

 ¿Cuáles son las principales preguntas que hacemos en una auditoría de configuración?

1. ¿Se ha hecho el cambio especificado en la OCI? ¿Se han incorporado modificaciones adicionales?
2. ¿Se ha llevado a cabo una revisión técnica formal para evaluar la corrección técnica?

3. ¿Se ha seguido el proceso del software y se han aplicado adecuadamente los estándares de ingeniería del software?
4. ¿Se han «resaltado» los cambios en el ECS? ¿Se han especificado la fecha del cambio y el autor? ¿Reflejan los cambios los atributos del objeto de Configuración?
5. ¿Se han seguido procedimientos de GCS para señalar el cambio, registrarlo y divulgarlo?
6. ¿Se han actualizado adecuadamente todos los ECSs relacionados?

En algunos casos, las preguntas de auditoría se incluyen en la revisión técnica formal. Sin embargo, cuando la GCS es una actividad formal, la auditoría de la GCS se lleva a cabo independientemente por el grupo de garantía de calidad.

## 9.7 INFORMES DE ESTADO

La *generación de informes de estado de la configuración* (a veces denominada *contabilidad de estado*) es una tarea de GCS que responde a las siguientes preguntas: (1) ¿Qué pasó? (2) ¿Quién lo hizo? (3) ¿Cuándo pasó? (4) ¿Qué más se vio afectado?

En la Figura 9.5 se ilustra el flujo de información para la generación de *informes de estado de la configuración* (IEC). Cada vez que se asigna una nueva identificación a un ECS o una identificación actualizada se genera una entrada en el IEC. Cada vez que la ACC aprueba un cambio (o sea, se expide una OCI), se genera una entrada en el IEC. Cada vez que se lleva a cabo una auditoría de configuración, los resultados aparecen como parte de una tarea de generación de un IEC. La salida del IEC se puede situar en una base de datos interactiva [TAY85] de forma que los encargados del desarrollo o del mantenimiento del software puedan acceder a la información de cambios por categorías clave. Además, se genera un IEC regularmente con intención de mantener a los gestores y a los profesionales al tanto de los cambios importantes.



*Desarrolle una «lista que se necesita conocer» para cada ECS y manténgala actualizada. Cuando se realice un cambio, asegúrese que se informa a todos los que están en la lista.*

La generación de informes de estado de la configuración desempeña un papel vital en el éxito del proyecto de desarrollo de software. Cuando aparece involucrada mucha gente es muy fácil que se dé el síndrome de que «la mano izquierda ignora lo que hace la mano derecha». Puede que dos programadores intenten modificar el mismo ECS con intenciones diferentes y conflictivas. Un equipo de ingeniería del software puede emplear meses de esfuerzo en construir un software a partir de unas especificaciones de hardware obsoletas. Puede que la persona que descubra los efectos secundarios de un cambio propuesto no esté enterada de que el cambio se está realizando. El IEC ayuda a eliminar esos problemas, mejorando la comunicación entre todas las personas involucradas.

## RESUMEN

La gestión de configuración del software es una actividad de protección que se aplica a lo largo de todo el proceso del software. La GCS identifica, controla, audita e informa de las modificaciones que invariablemente se dan al desarrollar el software una vez que ha sido distribuido a los clientes. Cualquier información producida como parte de la ingeniería del software se convierte en parte de una configuración del software. La configuración se organiza de tal forma que sea posible un control organizado de los cambios.

La configuración del software está compuesta por un conjunto de objetos interrelacionados, también denominados elementos de configuración del software, que se producen como resultado de alguna actividad de ingeniería del software. Además de los documentos, los programas y los datos, también se puede poner bajo control de configuración el entorno de desarrollo utilizado para crear el software.

Una vez que se ha desarrollado y revisado un objeto de configuración, se convierte en una línea base. Los

cambios sobre un objeto de línea base conducen a la creación de una nueva versión del objeto. La evolución de un programa se puede seguir examinando el historial de las revisiones de todos los objetos de su configuración. Los objetos básicos y los compuestos forman una asociación de objetos que refleja las variantes y las versiones creadas. El control de versiones es un conjunto de procedimientos y herramientas que se usan para gestionar el uso de los objetos.

El control de cambios es una actividad procedural que asegura la calidad y la consistencia a medi-

da que se realizan cambios en los objetos de la configuración. El proceso de control de cambios comienza con una petición de cambio, lleva a una decisión de proseguir o no con el cambio y culmina con una actualización controlada del ECS que se ha de cambiar.

La auditoría de configuración es una actividad de SQA que ayuda a asegurar que se mantiene la calidad durante la realización de los cambios. Los informes de estado proporcionan información sobre cada cambio a aquellos que tienen que estar informados.

## REFERENCIAS

- [ADA89] Adams, E., M. Honda y T. Miller, «Object Management in a CASE Environment», *Proc. 11<sup>th</sup> Intl. Conf. Software Engineering*, IEEE, Pittsburg, PA, Mayo 1989, pp. 154-163.
- [BAC98] Bach, J., «The Highs and Lows of Change Control», *Computer*, vol. 31, n.º 8, Agosto 1998, pp. 113-115.
- [BER80] Bersoff, E.H., V.D. Henderson y S. G. Siegel, *Software Configuration Management*, Prentice-Hall, 1980.
- [BRY80] Bryan, W., C. Chadbourne, y S. Siegel, *Software Configuration Management*, IEEE Computer Society Press, 1980.
- [CHO89] Choi, S. C., y W. Scacchi, «Assuring the Correctness of a Configured Software Description», *Proc. 2nd Intl. Workshop on Software Configuration Management*, ACM, Princeton, NJ, Octubre 1989, pp. 66-75.
- [CLE89] Clemm, G. M., «Replacing Version Control with Job Control», *Proc. 2nd Intl. Workshop on Software Configuration Management*, ACM, Princeton, NJ, Octubre 1989, pp. 162-169.
- [GUS89] Gustavsson, A., «Maintaining the Evaluation of Software Objects in an Integrated Environment», *Proc. 2nd Intl. Workshop on Software Configuration Management*, ACM, Princeton, NJ, Octubre 1989, pp. 114-117.
- [HAR89] Harter, R., «Configuration Management», *HP Professional*, vol. 3, n.º 6, Junio 1989.
- [IEE94] *Software Engineering Standards*, edición de 1994, IEEE Computer Society, 1994.
- [LIE89] Lie, A. *et al.*, «Change Oriented Versioning in a Software Engineering Database», *Proc. 2nd Intl. Workshop on Software Configuration Management*, ACM, Princeton, NJ, Octubre 1989, pp. 56-65.
- [NAR87] Narayanaswamy, K., y W. Scacchi, «Maintaining Configurations of Evolving Software Systems», *IEEE Trans. Software Engineering*, vol. SE-13, n.º 3, Marzo 1987, pp. 324-334.
- [REI89] Reichenberger, C., «Orthogonal Version Management», *Proc. 2nd Intl. Workshop on Software Configuration Management*, ACM, Princeton, NJ, Octubre 1989, pp. 137-140.
- [ROC75] Rochkind, M., «The Source Code Control System», *IEEE Trans. Software Engineering*, vol. SE-1, n.º 4, Diciembre 1975, pp. 364-370.
- [TAY85] Taylor, B., «A Database Approach to Configuration Management for Large Projects», *Proc. Conf. Software Maintenance-1985*, IEEE, Noviembre 1985, pp. 15-23.
- [TIC82] Tichy, W. F., «Design, Implementation and Evaluation of a Revision Control System», *Proc. 6<sup>th</sup> Intl. Conf. Software Engineering*, IEEE, Tokyo, Septiembre 1982, pp. 58-67.

## PROBLEMAS Y PUNTOS A CONSIDERAR

- 9.1.** ¿Por qué es cierta la primera ley de la ingeniería de sistemas? ¿Cómo afecta a nuestra percepción de los paradigmas de la ingeniería del software?
- 9.2.** Exponga las razones de la existencia de líneas base con sus propias palabras.
- 9.3.** Asuma que es el gestor de un pequeño proyecto. ¿Qué líneas base definiría para el proyecto y cómo las controlaría?
- 9.4.** Diseñe un sistema de base de datos que permita a un ingeniero del software guardar, obtener referencias de forma cruzada, buscar, actualizar, cambiar, etc., todos los elementos de la configuración del software importantes. ¿Cómo manejaría la base de datos de diferentes versiones de un mi-

mo programa? ¿Se manejaría de forma diferente el código fuente que la documentación? ¿Cómo se evitaría que dos programadores hicieran cambios diferentes sobre el mismo ECS al mismo tiempo?

- 9.5.** Investigue un poco sobre bases de datos orientadas a objetos y escriba un artículo que describa cómo se podrían usar en el contexto de la GCS.

- 9.6.** Utilice un modelo E-R (Capítulo 12) para describir las interrelaciones entre los ECS (objetos) de la Sección 9.1.2.

- 9.7.** Investigue sobre herramientas de GCS existentes y describa cómo implementan el control de versiones, de cambios de objetos de configuración.

**9.8.** Las relaciones «parte-de» e «interrelacionado» representan relaciones sencillas entre los objetos de configuración. Describa cinco relaciones adicionales que pudieran ser útiles en el contexto de la base de datos del proyecto.

**9.9.** Investigue sobre una herramienta de GCS existente y describa cómo implementa los mecanismos de control de versiones. Alternativamente, lea dos o tres de los artículos a los que se hace referencia en este capítulo e investigue en las estructuras de datos y los mecanismos que se usan para el control de versiones.

**9.10.** Utilizando la Figura 9.5 como guía, desarrolle un esquema de trabajo más detallado aún para el control de cambios. Describa el papel de la ACC y sugiera formatos para la petición de cambio, el informe de cambios e IEC.

**9.11.** Desarrolle una lista de comprobaciones que se pueda utilizar en las auditorías de configuración.

**9.12.** ¿Cuál es la diferencia entre una auditoría de GCS y una revisión técnica formal? ¿Se pueden juntar sus funciones en una sola revisión? ¿Cuáles son los pros y los contras?

## OTRAS LECTURAS Y FUENTES DE INFORMACIÓN

Uno de los pocos libros escritos sobre la GCS en los últimos años lo realizaron Brown et al. (*AntiPatterns and Patterns in Software Configuration Management*, Wiley, 1989).

Los autores tratan las cosas que no hay que hacer (anti patrones) cuando se implementa un proceso de GCS y entonces consideran sus remedios.

Lyon (*Practical CM: Best Configuration Management Practices for the 21<sup>st</sup> Century*, Raven Publishing, 1999) y Mikkelsen y Pherigo (*Practical Software Configuration Management: The Latenight Developer's Handbook*, Allyn & Bacon, 1997) proporcionan tutoriales prácticos importantes de GCS. Ben-Menachem (*Software Configuration Management Guidebook*, McGraw-Hill, 1994), Vacca (*Implementing a Successful Configuration Change Management Program*, I.S. Management Group, 1993), y Ayer y Patrinnostro (*Software Configuration Management*, McGraw-Hill, 1992) presentan correctas visiones para aquellos que todavía no se han introducido en la materia. Berlack (*Software Configuration Management*, Wiley, 1992), presenta un estudio útil de conceptos de la GCS, haciendo hincapié en la importancia del diccionario de datos (repository) y de las herramientas en la gestión del cambio. Babich [BAB86] es un tratado abreviado, aunque eficaz, de temas prácticos sobre la gestión de configuración del software.

Buckley (*Implementing Configuration Management*, IEEE Computer Society Press, 1993) estudia enfoques de gestión de configuración para todos los elementos de un sistema (hardware, software y firmware con unos detallados tratamientos

de las principales actividades de GC). Rawlings (*SCM for Network Development Environments*, McGraw-Hill, 1994) es el primer libro de GCS en tratar el asunto con un énfasis específico en el desarrollo de software en un entorno de red. Whitgift (*Methods and Tools for Software Configuration Management*, Wiley, 1991) contiene una razonable cobertura de todos los temas importantes de GCS, pero se distingue por su estudio del diccionario de datos y tratamiento de aspectos de entornos CASE. Arnold y Bohner (*Software Change Impact Analysis*, IEEE Computer Society Press, 1996) han editado una antología que estudia cómo analizar el impacto del cambio dentro de sistemas complejos basados en software.

Puesto que la GCS identifica y controla documentos de ingeniería del software, los libros de Nagle (*Handbook for Preparing Engineering Documents: From Concept to Completion*, IEEE, 1996), Watts (*Engineering Documentation Control Handbook: Configuration Management for Industry*, Noyes Publication, 1993), Ayer y Patrinnostro (*Documenting the Software Process*, McGraw-Hill, 1992) proporcionan un complemento con textos más centrados en la GCS. La edición de Marzo de 1999 de *Crosstalk* contiene varios artículos útiles de la GCS.

En Internet están disponibles una gran variedad de fuentes de información relacionadas con temas de gestión y de software. Se puede encontrar una lista actualizada con referencias a sitios (páginas) web que son relevantes para el Software en <http://www.pressman5.com>.

